EÖTVÖS LORÁND UNIVERSITY

THESES OF THE DOCTORAL DISSERTATION

Quality Aspects of TTCN-3 Based Test Systems

Author:
Kristóf SZABADOS

Supervisor:
Attila KOVÁCS, Dr. Habil.

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Eötvös Loránd University
Doctoral School of Informatics
Head: Prof. Dr. Erzsébet Csuhaj-Varjú
Information Systems Program
Head: Prof. Dr. András Benczúr

October 16, 2017

Chapter 1

Introduction

Since the first program was written the size and complexity of software systems have been growing constantly, together with the quality expectations against these systems. Nowadays the usage of software — developed by 11 million professional software developers [1] — belongs to the everyday life of our society. Software helps in navigating to destinations, communicating with other people, driving the production, distribution and consumption of energy resources. Software drives companies, trades on the markets, takes care of people's health. All of these systems must fulfill very strict (but different) quality restrictions. In the telecommunication area "five nines" (99.999%) availability allows only 5.26 minutes downtime per year — often including planned upgrades and maintenance.

Companies producing these systems perform several activities to ensure the required level of quality. They aim at automating tests, while managing their size and complexity, which clearly grows together with the tested systems. In the telecommunication area this pressure facilitated the ETSI[1] to develop TTCN-3[2], a scripting language used in testing the conformance of communicating systems to standards and for specification of test infrastructure interfaces that glue abstract test scripts with specific communication environments.

1.1 Motivation

Although tests evolved to be large and complex together with their own standardized language, the internal quality, complexity, structure and evolution of test scripts is not yet a well studied subject. Tassey [2] found that inadequate software testing infrastructure may cost between 22.2 and 59.5 billion USD annually (in the U.S. only).

Companies want to minimize the duration of the final testing period a product has to go through before reaching the market. This is best achieved by developing the tests before or in parallel with the product itself. Once the first market ready product version is ready it would just need to be tested, by all of the tests. In practice however, this means that testers sometimes have to write end-to-end tests months or years before the actual system to be tested is fully available. Checking the correctness of all of the tests before the product is ready might not be possible and is done under heavy business pressure once it is (testing viewed as being the last "roadblock" before market availability).

[1] European Telecommunications Standards Institute
[2] Testing and Test Control Notation - 3

Architects of such test systems had to work for several years with limited tool support, without being able to see what the actual architecture of their test system looks like. Generic graph layout algorithms — freely available to be used to analyze and visualize architecture — don't fit well into daily operations in the industry. Detecting architectural issues or telling what action should be done next is hard using generic layouts. In the current situation at the industrial scale most layout algorithms can take several seconds to calculate [3], making interactive work impossible. It is also not clear who they target: system architects do not have much time to look into the details, and testers might lack the high-level view of the systems.

Management issues don't stop with the issues of testers and test system architects. As large scale test systems are a recent phenomena, there is little information on how to manage them. It is not really known what effect different management practices and organizational changes can have, how these systems evolve, how to handle complexity. There are only anecdotal evidences on how to hire for development and testing jobs, and even those might not be applicable to hire for test automation jobs (where the employee should "think" like a tester, and be able to write complex automated tests at the same time).

The TTCN-3 language itself is still under construction and changes rapidly.

1.2 Objectives and methods

The objective of this work is to research tools, methods, knowledge that can support the creation of high quality large scale test systems.

To support testers with quality checks, we reviewed already existing bug databases, tools and literature to find rules that can be applied to TTCN-3 and checked automatically. Using the Delphi [12] method we derived real life correction costs for these issues.

We analyzed the architecture of large scale test systems using graph properties and designed 2 new visual layouts that fit better into System Architects daily work.

We did a longitudinal study on the development of a large scale test system to understand how test systems evolve in industrial settings.

We also surveyed a large group of people (IT professionals working at companies present in Hungary) to understand how IT professionals knowledge differs having various roles (manager, developer, tester, technical writer), how they gain new knowledge, how they vary in thinking about their processes and anti-patterns in software development

To have a wide reach in industrial and standardization setting, we extended the open-source tool Titan [4][3]. All functionality developed for our research is freely available as part of the Titan tool under the name Titanium.

[3] Titan is an open source TTCN-3 test toolset used in Ericsson for functional and load testing by more than 4000 internal users

Chapter 2

Results

We look at *tests as software products*, we view TTCN-3 as a *programming language*. We analyze software products written in TTCN-3 to see how they compare to "normal" software products.

2.1 The quality of tests

We used a 3 sources to identify situations in TTCN-3 code that might indicate quality problems.

We have reviewed the databases of source code review documents, errors and problems found in released products, created at our industry partner. These records contained code quality issues which may became show stoppers in any TTCN-3 project's life cycle.

We have also checked the rules of PMD [5], FxCop [6], Checkstyle [7], FindBugs [8], xUnit Patterns [9], Martin Fowler's book on refactoring [10] and TRex [11] for static analyzer rules that can be used in testing and in particular for the TTCN-3 language. We found that only a few rules were applicable to our purposes.

We analyzed the semantic checking and code generation algorithms of Titan for situations which result in low quality or badly performing code.

Based on this work we created the list of code smell rules we found to be applicable to TTCN-3. Each rule was discussed with experienced, professional TTCN-3 experts from our industry partner and categorized into the classes which it most likely belongs to, according to the ISO/IEC 9126 and ISO/IEC 25010 quality models. Most likely means that more than 66% of the review meeting members agreed.

Thesis 1: I defined and analyzed TTCN-3 code smells, classified them according to international software quality standards and presented a method for qualifying TTCN-3 based test systems.

With the implementation of a subset of these code smells we analyzed all test systems which were available at `www.ttcn-3.org` in January 2014 and some industrial test systems at our industry partner. The webpage lists links to test suites provided by 2 different standardization organizations: ETSI and 3GPP[1].

During our analysis we have found several quality problems in the analyzed software packages. Showing the need for an automatic code smell analyzer.

[1] 3rd Generation Partnership Project

Thesis 2: I found several internal quality issues in both industrial and standardized TTCN-3 test suites.

Applying the Delphi method, we collected estimates from experts in the field of test software engineering at our industry partner on how long the correction of a single instance of a given code smell type would take. The team consisted of a test system architect, test system developers and engineers working in maintenance & support.

Applying the estimated correction times we found that standardized test suites have substantial technical debt. In the average difficulty case[2] the technical debt of the projects can be measured on 1000 Mhr base meaning several man-years of technical debt.

Thesis 3: I analyzed and assessed the costs of correcting the found internal quality issues of the defined code smell items.

Publications belonging to these theses: [19, 16, 17, 18, 26]

2.2 The architecture of test systems

We analyzed the module structure of eleven TTCN-3 based test projects. Some of these projects were standardized, some were industrial.

We measured for each module in each project how many others they import, and how many times they were imported by other modules.

According to our measurements on bigger projects, in-degree values follow a logarithmic trend, out-degree values follow a power law trend.

In case of TTCN-3 the diameter of the module importation graph (the longest path from the set of the shortest paths between any two nodes in the graph) seems to be a logarithmic function of the number of modules present in the project. This is in line with previous observations [13] on small-world and scale-free networks.

Thesis 4: I observed that large scale TTCN-3 test suites show small-world properties and seem to converge to scale-free.

We implemented two layout algorithms using JUNG [14] to visualize the architecture and analyzed all test suites (40) publicly available at ETSI's official TTCN-3 homepage `www.ttcn-3.org` in 2016. In these test suites we found several architectural problems.

- In several cases we found files independent from the rest of the test suite.
- Many test suites had top level files, which might not be needed.
- We found one test suite with import cycles among modules.
- Several test suites had import cycles among their folders.

According to our survey test system architects found our tool useful. Two architects corrected 57% of the reported circular dependencies resulting in a 3% improvement in the build time of the whole system.

[2] All detected code smell instances assumed to require average amount of work to solve

Thesis 5: Based on my analysis I was able to show that TTCN-3 test systems contain issues on architectural level and my visualization solution makes it easier to detect these issues comparing to other available solutions.

Publications belonging to these theses: [15, 20]

2.3 The evolution of test systems

We have studied the 5 year long development of two test systems developed and used at our industry partner.

For each day in the investigated range we checked out the source code in the state it was at midnight and measured the number of code smells present.

Correlating the trends observed for the different measured code smells and comparing these measured trends with important events in the project's history we made the following observations:

- The number of measured code smells was affected by the merging of two test systems.
- The number of measured code smells was not affected by the introduction of continuous integration, by the introduction of tool support itself, by the different development methodologies, by changing technical, line and project leaders of the projects.
- Code smells in the observed test system followed predictable patterns during the system's evolution.

Thesis 6: I observed that the internal quality evolution of the examined TTCN-3 test systems follows a predictable pattern similar to that of programming languages and projects.

Publications belonging to these thesis: [22]

2.4 Human Side of Quality

We surveyed individuals working in software development projects to understand how the knowledge of IT employees differs having various roles (manager, developer, tester, technical writer), how they gain new knowledge, how they vary in thinking about their processes and anti-patterns in software development.

Based on the 456 responses received we have seen that software is mostly developed in large companies, by people with little experience (mostly less than 2 years) in their role. According to our survey most respondents turn to the Internet, colleagues or books to learn new skills. Among the respondents both developer and tester mindsets are recognized to be very important in software development projects.

According to our observations people in different roles think alike and/or follow similar processes. The answers for thinking and processes was the most similar in the case of developers and testers.

Thesis 7: I observed that the mindset of testers and developers is similar. To be more precise I showed that from human aspects regarding the internal quality a test project is very similar to a software project.

Publications belonging to these thesis: [21, 25]

2.5 Summary

Although tests have evolved to be large and complex together with the production software systems they test, their internal quality, complexity, structure and evolution is not a well studied subject.

In this work we looked at *tests as software products*, we viewed TTCN-3 as a *programming language*. We analyzed software products written in TTCN-3 to see how they compare to "normal" software products.

We were among the first drawing attention to the quality issues of industrial and standardized test systems.

To the best of our knowledge we were the first to point out that test and production software systems evolve similarly and adopt similar architectural properties (or converge to them).

Our survey of IT professionals found that the thinking of developers and testers is very similar. According to our knowledge we were the first to investigate this side of the human aspect of IT.

Bibliography

[1] A. Avram: *IDC Study: How Many Software Developers Are Out There?*, 2014, https://www.infoq.com/news/2014/01/IDC-software-developers, last visited: January, 2017.

[2] G. Tassey: *The Economic Impacts of Inadequate Infrastructure for Software Testing*, 2002, Final report, Prepared by RTI for the National Institute of Standards and Technology (NIST), https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf, last visited: January, 2017.

[3] S. Hachul, M. Junger: *Large-Graph Layout Algorithms at Work: An Experimental Study*, Journal of Graph Algorithms and Applications, Vol. 11, No. 2, 2007, pp. 345–369.
DOI: 10.7155/jgaa.00150

[4] TITAN, https://projects.eclipse.org/proposals/titan, last visited: January, 2017.

[5] *PMD*, http://pmd.sourceforge.net, last visited: January 2017.

[6] *FxCop*, http://msdn.microsoft.com, last visited: January 2017.

[7] *Checkstyle*, http://checkstyle.sourceforger.net, last visited: January 2017.

[8] *FindBugs*, http://findbugs.sourceforge.net, last visited: January 2017.

[9] G. Meszaros: *xUnit Test Patterns: Refactoring Test Code*, Addison-Wesley,
ISBN-10: 0131495054, ISBN-13: 978-0131495050

[10] M. Fowler: *Refactoring: Improving the Design of Existing Code*, 1999, Addison-Wesley Longman Publishing Co. Inc., Boston, USA.
ISBN-10: 0-201-48567-2, ISBN-13: 978-0201485677

[11] *TRex*, http://www.trex.informatik.uni-goettingen.de/trac, last visited: January 2017.

[12] L. Helmer: *Analysis of the future: The Delphi method*, RAND Corporation, 1967, http://www.rand.org/pubs/papers/P3558.html, last visited: January 2017.

[13] R. Cohen and D. Hevlin: *Scale-Free Networks are Ultrasmall*, Physical Review Letters, Vol. 90/5, 058701, 2003. https://doi.org/10.1103/PhysRevLett.90.058701, last visited: January 2017.

[14] Java Universal Network/Graph Framework,
http://jung.sourceforge.net/, last visited: January 2017.

Own papers, conference talks, posters

[15] K. Szabados: *Structural Analysis of Large TTCN-3 Projects*, In Proceeding Of Testing of Software and Communication Systems, 21st IFIP WG 6.1 International Conference, TESTCOM 2009 and 9th International Workshop, FATES 2009, Eindhoven, The Netherlands, November 2-4, Lecture Notes in Computer Science: Testing of Software and Communication Systems, Springer, 2009, pp. 241–246.

ISBN: 978-3-642-05030-5, DOI: 10.1007/978-3-642-05031-2_19

[16] K. Szabados and A. Kovács: *Test software quality issues and connections to international standards*, Acta Universitatis Sapientiae, Informatica, 5/1, 2013, pp. 77–102.

DOI: 10.2478/ausi-2014-0006

[17] K. Szabados and A. Kovács: *Advanced TTCN-3 Test Suite validation with Titan*, In Proceedings of the 9th International Conference on Applied Informatics, Vol. 2, 2014, pp. 273–281.

DOI: 10.14794/ICAI.9.2014.2.273

[18] K. Szabados and A. Kovács, *Technical debt of standardized test software*, IEEE 7th International Workshop on Managing Technical Debt (MTD), Bremen, 2015, pp. 57–60.

DOI: 10.1109/MTD.2015.7332626

[19] K. Szabados and A. Kovács, *Up-to-date list of code smells*, http://compalg.inf.elte.hu/~attila/TestingAtScale.htm, last visited: January, 2017.

[20] K. Szabados, A. Kovács, G. Jenei and D. Góbor: *Titanium: Visualization of TTCN-3 system architecture*, IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2016, pp. 7–11.

DOI: 10.1109/AQTR.2016.7501275

[21] K. Szabados and A. Kovács: *Knowledge and mindset in software development – how developers, testers, technical writers and managers differ – a survey*, 11th Joint Conference on Mathematics and Computer Science (MACS), Eger, Hungary, 2016. State: Accepted for publication.

[22] K. Szabados and A. Kovács: *Internal quality evolution of a large test system – an industrial study*, Acta Universitatis Sapientiae, Informatica, 8/2, 2016, 216–240.

[23] K. Szabados: *Creating an efficient and incremental IDE for TTCN-3*, 10th Joint Conference on Mathematics and Computer Science, Cluj-Napoca, In Studia Universitatis Babes-Bolyai, Informatica, Volume LX, Number 1, 2015, pp. 5–18.

[24] K. Szabados and A. Kovács: *Developing and Testing at Large Scale*, 5th Annual International Conference of the Hungarian Software Testing Forum (HUSTEF), Budapest, Hungary, 2015. (Talk)

[25] K. Szabados: *Thinking/mindset of testers is the closest to that of developers*, 6th International Conference of the Hungarian Software Testing Forum (HUSTEF), Budapest, Hungary, 2016. (Poster)

[26] K. Szabados, Gy. Réthy: *Test Software Quality Through Software Metrics*, 1st User Conference on Advanced Automated Testing (UCAAT 2013), Paris, 2013. (Poster)

[27] K. Szabados, A. Kovács: *Test systems, software systems. Is there a difference?*, 3rd User Conference on Advanced Automated Testing (UCAAT 2015), ETSI, Sophia Antipolis, 2015. (Talk)